# KTH Challenge

*KTH Challenge 2021*

*May 8, 2021*

Problems

Do not open before the contest has started.

This page is intentionally left (almost) blank.

# Problem A
# Alto Singing

Sheet music, public domain

Anton is a famous choir singer in the *Kongliga Teknologkören*, the KTH choir. After years of careful political maneuvering, he managed to become the chairman of the choir. As such he has absolute power of what the choir does – including what pieces to rehearse.

Anton has chosen a particular song he likes very much that he wants the choir to sing. There is only a slight problem – the song is not at all suited for his particular vocal range. Anton has the *alto* vocal range, and wants to make sure that the most important part of the piece is perfect for his voice.

The piece is currently written down using the $12$ standard tones, in ascending order C, C#, D, D#, E, F, F#, G, G#, A, A#, B, each followed by an integer denoting the octave the tone appears in. The lowest octave is $1$. Immediately after the B tone of octave $i$, the C tone of octave $i + 1$ follows.

Now, Anton wants to *transpose* the piece so that all tones in the part are within his vocal range, which is given by the lowest and highest tone he can sing. Transposing means that all tones are shifted either up or down by some fixed number of tones. For example, C#4 (C# in the fourth octave) transposed $5$ tones down would be G#3. Furthermore, Anton is not very good at reading sheet music. Among all valid transpositions, he is only interested in the ones that minimize the number of tones with accidentals (that is, those with a # sign)[1].

Given the piece and Anton's vocal range, can you determine how many such transpositions there are?

## Input

The first line of input contains the number of tones $n$ ($1 \leq n \leq 1000$) in the piece. Then follows a line containing two tones, the lowest and the highest tone that Anton can sing (in that order).

The third and final line contains the tones that Anton's piece consists of. The same tone may appear multiple times in the piece. It is guaranteed that Anton can sing at least one transposition of the piece.

Each tone is written as one of the $12$ tones C, C#, D, D#, E, F, F#, G, G#, A, A#, B followed immediately by its octave without any space in between. Only octaves between $1$ and $10^9$ are used[2].

## Output

Output a single line with two integers – the minimum number of accidentals (notes with #) among all transpositions, and the number of transpositions that keeps the piece in Anton's vocal range and has the minimum number of accidentals. If the piece is already in Anton's vocal range, the transposition of $0$ tones should be counted as well (if that minimizes accidentals).

[1]For those of you who know music, Anton finds writing down a key for the piece other than C major even more confusing, since he must then remember what tones should be raised or lowered throughout the piece.

[2]Possibly only after centuries of vocal exercises.

**Sample Input 1**

```
14
F3 F5
C4 C4 G4 G4 A4 A4 G4 F4 F4 E4 E4 D4 D4 C4
```

**Sample Output 1**

```
0 3
```

**Sample Input 2**

```
1
C#1 A#1
C1000000000
```

**Sample Output 2**

```
0 5
```

**Sample Input 3**

```
3
F3 F5
F3 F#3 F5
```

**Sample Output 3**

```
1 1
```

# Problem B
# Bus Lines

After many years without any public transport, the town Krockholm will finally get a network of bus lines. The plans are still on the drawing board, but it has been decided that there shall be $n$ stations labelled $1$ to $n$, and $m$ bus lines where each line connects two stations. The only thing remaining is to decide which pairs of stations should be connected. One important requirement is that it should be possible to get from any station to any other. In addition to this, someone had the brilliant idea that the bus lines should be labelled by the sum of their endpoints. This means that all of these sums must be different.

You are given two integers $n$ and $m$. Construct a graph with $m$ edges and $n$ vertices labelled $1$ to $n$, such that:

1. The graph is connected.
2. The sums of edge endpoints are distinct.

## Input

The input consists of a single line containing two integers $n$ and $m$ ($2 \leq n \leq 100, 1 \leq m \leq 10^4$).

## Output

If it is not possible to construct a graph with the given properties, print "`-1`". Otherwise, print $m$ lines where the $i$'th line contains two integers $a_i$, $b_i$, the endpoints of the $i$'th edge. If there are many possible solutions, any one of them will be accepted.

| Sample Input 1 | Sample Output 1 |
|---|---|
| `4 4` | `2 1`<br>`2 3`<br>`4 3`<br>`4 2` |

| Sample Input 2 | Sample Output 2 |
|---|---|
| `10 100` | `-1` |

| Sample Input 3 | Sample Output 3 |
|---|---|
| `10 1` | `-1` |

This page is intentionally left (almost) blank.

# Problem C
# Costly Contest

The company Mindsight is holding a programming contest for $n$ contestants of varying ages. It has been decided that the contest will be separated into $k$ age divisions. The duration of the contest will be $t$ minutes, the same for all divisions. Mindsight has created a pool of $m$ available problems for use in the contest, and since all divisions will compete at the same time, the same problems can be used in multiple divisions without any issues.

In each division, the participant that solves the largest number of problems gets a prize, and in case of a tie (same number of problems solved) everyone tied for first place gets a prize. In particular, if no one in a division solves any problems, then everyone in that division gets a prize.

Mindsight has now come to the horrible realization that with these rules it is possible that all participants win a prize. This could bankrupt the company! So the company has enlisted a team of experts to help resolve the situation.

The expert group analyzed the data in depth. For each of the $n$ participants, their skill level was quantified: for the $i$'th participant, a *slowness factor* $s_i$ was determined. Then, for each of the $m$ problems available, its difficulty was quantified: for the $j$'th problem, a *difficulty rating* $d_j$ was assigned. The experts predict that the time it takes for participant $i$ to solve problem $j$ is $s_i \cdot d_j$ minutes. Furthermore participants cannot work on multiple problems in parallel so the time it takes to solve multiple problem is the sum of times of solving the individual problems. It is also well-established that participants always solve problems in increasing order of difficulty, starting with the easiest problem.

Now it is up to you, the underpaid intern, to configure the divisions so as to minimize the number of awarded prizes. Your task is to partition the $n$ participants into $k$ non-empty age divisions, and for each age division choose a non-empty subset of the $m$ available problems to use in that division. Each division must correspond to a *contiguous* age range of participants (e.g. a division *cannot* be "20-25 or 30-35 years old"). Recall that the same problem may be used in multiple divisions.

## Input

The first line of input contains four integers $n, m, k, t$ ($1 \leq n \leq 10^5$, $1 \leq m \leq 100$, $1 \leq k \leq n$, $1 \leq t \leq 10^5$) – the number of participants $n$, the number of available problems $m$, the number of age divisions $k$, and the duration of the contest $t$. The second line contains $n$ integers $s_1, \ldots, s_n$ the slowness factors of the participants ($1 \leq s_i \leq 10^5$). The participants are ordered by age, and you can assume no two participants have the same exact age. The third line contains $m$ integers $d_1, \ldots, d_m$ the difficulty ratings of the problems ($1 \leq d_j \leq 10^5$).

## Output

Output a single integer, the minimum number of prize winners.

| Sample Input 1 | Sample Output 1 |
|---|---|
| 4 3 2 10<br>2 4 3 4<br>11 3 6 | 2 |

**Sample Input 2**

```
4 3 1 10
4 2 2 6
2 4 4
```

**Sample Output 2**

```
2
```

# Problem D
## DEX Save

Purple d20, public domain

Oh no! After months of playing a campaign in the table-top role-playing game Bungalows & Banshees, your character accidentally triggered a trap and was squished under a gigantic rolling boulder. If you had only succeeded in that dexterity saving throw to dodge out of the way of the incoming boulder of doom...

Always one to dwell on the past, you begin to wonder what your chances actually were of succeeding with the saving throw, and discover that it is not immediately obvious – you had that bardic inspiration giving you a d6 bonus, but then you had disadvantage on the roll due to being intoxicated, but on the other hand your dexterity save modifier was pretty high...

A basic dexterity saving throw with difficulty $d$ is performed in the following way: first, a d20 (i.e., a 20-sided die) is rolled. If the result of the roll is $1$ the result is immediate failure, and if the result is $20$ the result is immediate success. Otherwise, the character's DEX save modifier $m$ is added to the die result. If the sum is at least the difficulty $d$, the result is a success, otherwise it is a failure.

There are two extensions to this. First, a roll may be made with either advantage or disadvantage. In these cases, two d20s are rolled instead of one, but then only the highest (for advantage) or lowest (for disadvantage) is kept and the result is then computed as in the basic case. Second, the roll may have additional bonuses or penalty in the form of additional dice that are rolled and then added to or subtracted from the d20 result before comparing it to $d$ (but these bonuses/penalties are only rolled once regardless of advantage or disadvantage).

Write a program which, given the data of a saving throw (its difficulty, the DEX save modifier, advantage/disadvantage status, and additional bonus dice) computes the probability that the saving throw will succeed.

## Input

The input consists of three lines. The first line contains two integers $d$ and $m$ ($0 \leq d \leq 30$, $-10 \leq m \leq 10$), the difficulty of the roll and the DEX save modifier of the character. The second line contains one word indicating if the roll has advantage or disadvantage. This word is either "`straight`" (for a straight roll with neither advantage or disadvantage), "`advantage`", or "`disadvantage`". Finally, the third line starts with an integer $k$ ($0 \leq k \leq 5$) indicating the number of additional bonus/penalty dice. This is followed by $k$ dice descriptions, each of the form "`[+-]d`$x$" ($3 \leq x \leq 10$ is an integer) indicating that we add (if '+') or subtract (if '−') the outcome of an $x$-sided die to the result.

## Output

Output a single number, the probability that a dexterity saving throw with the given parameters will succeed. This number should be given with an absolute error of at most $10^{-6}$.

| Sample Input 1 | Sample Output 1 |
|---|---|
| 12 4<br>straight<br>0 | 0.65 |

**Sample Input 2**

```
10 3
advantage
0
```

**Sample Output 2**

```
0.91
```

**Sample Input 3**

```
20 7
disadvantage
2 +d6 -d4
```

**Sample Output 3**

```
0.212916667
```

# Problem E
# Even Electricity

A town gets all its electricity from solar power and a dam that provides hydroelectric power. These sources of energy are not always the most reliable, since the amount of sun and water can vary a lot from day to day. Luckily, the dam has a large reservoir that can store water in order to even out the supply of electricity from day to day.

You will be given information about the $n$ coming days. On the $i$'th day, the solar power station generates $s_i$ units of electricity. At the same time, $w_i$ units of water will flow into the reservoir. Each unit of water can be converted into one unit of electricity, and you can decide how much you want to convert and how much should be left in the reservoir. However, the reservoir can only hold $r$ units of water, so at the end of the day the amount of water in the reservoir cannot be larger than $r$. In the beginning of the first day the reservoir is empty.

On the first day, there is no water currently stored in the reservoir. At the end of the last day, the dam will undergo maintenance and *must be empty*. In other words, all the water that flows into the dam during the $n$ days must at some point be converted to electricity.

Let $e_i$ be the amount of electricity produced on the $i$'th day. Note that $e_i = s_i + r_i$ where $r_i$ is how much water from the reservoir you converted on the $i$'th day. Your task is to find the minimum possible value of $\max_i(e_i) - \min_i(e_i)$.

## Input

The first line contains two integers $n$ and $r$ ($1 \leq n \leq 10^5$, $1 \leq r \leq 10^9$), the number of days and the amount of water the reservoir can hold.

The following $n$ lines each contain two integers $s_i$ and $w_i$ ($0 \leq s_i, w_i \leq 10^9$), the amount of electricity generated by the solar power station, and the amount of water that flowed into the reservoir on the $i$'th day.

## Output

Output the minimum possible value of $\max_i(e_i) - \min_i(e_i)$.

**Sample Input 1**

```
3 1
4 2
2 0
5 1
```

**Sample Output 1**

```
3
```

This page is intentionally left (almost) blank.

# Problem F
# Forgotten Homework

Yesterday was a fun first day of University for Alice. She met so many new people, and there were so many activities to do! But her new teacher was super strict. It was her first day of university, and the teacher still gave them a lecture on linear algebra and matrix multiplication. And not only that, in order to test the new students, the teacher also gave them the following homework due to the next day.

> *You are given an $n \times n$ matrix $A$ and two indices $i$ and $j$. Calculate the sequence $A^1(i,j), A^2(i,j), ..., A^{2n}(i,j)$ by hand. In other words, the element on row $i$ and column $j$ in each of the matrices $A^1, A^2, ..., A^{2n}$. Since these numbers can become very large, you should do all calculations modulo $10^9 + 7$.*

Alice, wanting to be a perfect student, definitely did not want to fail this task. So she stayed up all night yesterday doing matrix calculations. But oh no! In the morning, on her way to university, she realizes that she forgot to write down the last number $A^{2n}(i,j)$. And now she has no time left to fix her mistake.

Help Alice calculate the number she is missing!

## Input

The first line contains three integers $n$ and $i$ and $j$ ($1 \leq n \leq 3000$, $1 \leq i \leq n$, $1 \leq j \leq n$), given to Alice by her teacher. The second line contains the incomplete sequence of $2n - 1$ integers $A^1(i,j), A^2(i,j), ..., A^{2n-1}(i,j)$ calculated by Alice ($0 \leq A^k(i,j) < 10^9 + 7$ for $k = 1, 2, \ldots, 2n-1$). You may assume that Alice made no mistakes calculating the incomplete sequence. The remaining $n$ lines of input each contain $n$ integers, giving the matrix $A$. The $c$'th number in the $r$'th of these lines is the entry $A(r,c)$ in row $r$ column $c$ of $A$ ($0 \leq A(r,c) < 10^9 + 7$).

## Output

Output the value of $A^{2n}(i,j)$ modulo $10^9 + 7$.

**Sample Input 1**

```
3 1 1
0 2 0 4 0
0 1 1
1 0 0
1 0 0
```

**Sample Output 1**

```
8
```

**Sample Input 2**

```
1 1 1
2
2
```

**Sample Output 2**

```
4
```

This page is intentionally left (almost) blank.

# Problem G
## Guessing Circle

Alf and Beata were two young adults living together a long, long time ago, before you could spend all your afternoons competing in programming. Their lives were thus much more boring than those of today's young adults. How could you even survive back then, you might ask yourself. The answer is simple: you write down numbers on pieces of paper! Our two cohabitants loved writing integers on papers, and often had huge piles of them each afternoon. To avoid filling their entire living room with integers, Beata challenged her friend to a game every evening to determine who should take the trash out – the Guessing Circle game.

The Guessing Circle game is played by two players (in our case, Alf and Beata) using a large circle of $n$ pieces of papers, each paper labelled with some integer. Alf starts by choosing an integer $x$ that appears on some piece of paper. Beata then tries to figure out what this integer is by asking a series of questions. In each question, Beata picks an integer $y$ that appears on a piece of paper and asks if $y$ is closest to $x$ on the circle when going clockwise or counter-clockwise (measured in the number of pieces of paper between them). If both directions give the same distance, for instance if $y = x$, Alf can choose which one of the two possible answers to provide.

They had initially agreed that no two pieces of paper may have the same integer written on them, but Alf found this hugely unfair – it was quite easy for Beata to figure out $x$. Instead, he suggested a variant where some integers can appear multiple times in the circle. When providing the answer to a question $y$ from Beata, he is instead allowed to choose *any* pair of papers on which $x$ and $y$ appear, and give his answer for these two papers.

Beata reluctantly agreed to play the new variant, as long as Alf promises to choose an integer in the circle such that Beata can eventually figure it out. Knowing which these integers are turned out to be quite a tricky task for Alf, and he often had to spend hours before the game proving that Beata would be able to deduce which number he had chosen. Write a program to help Alf determine which numbers he can choose.

## Input

The first line of integers contains $n$ ($2 \leq n \leq 15\,000$), the number of pieces of paper in the circle. The next line contains $n$ integers, the integers written on the pieces of paper, each between $1$ and $15\,000$. They are given clockwise in the order they appear on the circle, and are not necessarily unique.

## Output

Output all integers $x$ that Alf can choose in the game such that given enough guesses, Beata can uniquely determine the value of $x$. List these values in increasing order. If there is no such integer $x$, output "`none`".

**Sample Input 1**

```
3
1 2 3
```

**Sample Output 1**

```
1
2
3
```

**Sample Input 2**

```
3
1 1 2
```

**Sample Output 2**

```
none
```

**Sample Input 3**

```
4
1 2 1 3
```

**Sample Output 3**

```
none
```

**Sample Input 4**

```
5
1 2 3 4 1
```

**Sample Output 4**

```
1
```